

Abstract Base Class: **BankAccount**

javascript

CopyEdit

```
class BankAccount {
  constructor(accountNumber, owner, balance = 0) {
    if (new.target === BankAccount) {
      throw new Error("Cannot instantiate abstract class directly");
    }
    this.accountNumber = accountNumber;
    this.owner = owner;
    this.balance = balance;
  }
}
```

- **BankAccount Class:** This is an abstract class that cannot be instantiated directly.
- **Constructor (**constructor**):** Initializes the **accountNumber**, **owner**, and **balance** properties of the account. If you try to create an instance of **BankAccount** directly, it will throw an error (as this class is abstract).

javascript

CopyEdit

```
  deposit(amount) {
    throw new Error("Method 'deposit()' must be implemented in subclass");
  }

  withdraw(amount) {
    throw new Error("Method 'withdraw()' must be implemented in subclass");
  }
}
```

- **Abstract Methods (**deposit** and **withdraw**):** These are placeholder methods meant to be implemented in subclasses. If they are not implemented in subclasses, they will throw an error.

javascript

CopyEdit

```
  getBalance() {
    return this.balance;
  }
}
```

```

    }

    getAccountInfo() {
        return {
            accountNumber: this.accountNumber,
            owner: this.owner,
            balance: this.balance
        };
    }
}

```

- **getBalance Method:** Returns the balance of the account.
- **getAccountInfo Method:** Returns an object with account details (accountNumber, owner, and balance).

SavingsAccount Class (Subclass of BankAccount)

javascript
CopyEdit

```

class SavingsAccount extends BankAccount {
    constructor(accountNumber, owner, balance = 0) {
        super(accountNumber, owner, balance);
        this.interestRate = 0.03; // 3% interest
    }
}

```

- **SavingsAccount Class:** Inherits from BankAccount and adds functionality specific to savings accounts.
- **Constructor (constructor):** Initializes the interestRate property (set to 3%) in addition to the inherited properties (accountNumber, owner, balance).

javascript
CopyEdit

```

deposit(amount) {
    if (amount > 0) {
        this.balance += amount;
        console.log(`₹${amount} deposited to Savings. Balance: ₹${this.balance}`);
    }
}

```

```
}  
}
```

- **deposit Method:** If a valid amount is provided (greater than 0), it adds the `amount` to the `balance` and logs the deposit.

javascript

CopyEdit

```
withdraw(amount) {  
    if (amount <= this.balance) {  
        this.balance -= amount;  
        console.log(`₹${amount} withdrawn from Savings. Balance:  
₹${this.balance}`);  
    } else {  
        console.log("Insufficient balance in Savings.");  
    }  
}
```

- **withdraw Method:** If the withdrawal amount is less than or equal to the balance, it subtracts that amount from the `balance`. Otherwise, it logs an "Insufficient balance" message.

javascript

CopyEdit

```
applyInterest() {  
    const interest = this.balance * this.interestRate;  
    this.balance += interest;  
    console.log(`Interest ₹${interest.toFixed(2)} added. New  
Balance: ₹${this.balance}`);  
}
```

- **applyInterest Method:** Calculates interest based on the current balance and the `interestRate`, then adds it to the `balance`. It logs the interest added and the updated balance.

CurrentAccount Class (Subclass of BankAccount)

javascript

CopyEdit

```
class CurrentAccount extends BankAccount {
  constructor(accountNumber, owner, balance = 0) {
    super(accountNumber, owner, balance);
    this.overdraftLimit = 10000; // Overdraft limit
  }
}
```

- **CurrentAccount Class:** Inherits from `BankAccount` and adds functionality specific to current accounts.
- **Constructor (`constructor`):** Initializes the `overdraftLimit` property (set to ₹10,000) in addition to the inherited properties (`accountNumber`, `owner`, `balance`).

javascript

CopyEdit

```
deposit(amount) {
  if (amount > 0) {
    this.balance += amount;
    console.log(`₹${amount} deposited to Current. Balance: ₹${this.balance}`);
  }
}
```

- **deposit Method:** If the deposit amount is valid (greater than 0), it adds the `amount` to the `balance` and logs the deposit.

javascript

CopyEdit

```
withdraw(amount) {
  if (amount <= this.balance + this.overdraftLimit) {
    this.balance -= amount;
    console.log(`₹${amount} withdrawn from Current. Balance: ₹${this.balance}`);
  } else {
    console.log("Overdraft limit exceeded.");
  }
}
```

- **withdraw Method:** This method allows withdrawals within the **balance** plus the **overdraftLimit**. If the withdrawal amount exceeds the sum of the balance and the overdraft limit, it logs "Overdraft limit exceeded."
-

Testing the Code

javascript

CopyEdit

```
const savings = new SavingsAccount("111", "Aman", 5000);
savings.deposit(1000);
savings.withdraw(2000);
savings.applyInterest();
console.log("Savings Account Info:", savings.getAccountInfo());
```

- A new **SavingsAccount** instance is created with account number **111**, owner **Aman**, and a balance of ₹5000.
- **Deposit:** ₹1000 is deposited into the savings account, increasing the balance to ₹6000.
- **Withdraw:** ₹2000 is withdrawn, leaving the balance at ₹4000.
- **Apply Interest:** The interest (3% of ₹4000) is added to the balance ($₹4000 * 0.03 = ₹120$), updating the balance to ₹4120.
- Finally, the account info is logged, showing the account details and the updated balance.

javascript

CopyEdit

```
const current = new CurrentAccount("222", "Neha", 2000);
current.deposit(3000);
current.withdraw(6000);
current.withdraw(8000); // Should hit overdraft limit
console.log("Current Account Info:", current.getAccountInfo());
```

- A new **CurrentAccount** instance is created with account number **222**, owner **Neha**, and a balance of ₹2000.
- **Deposit:** ₹3000 is deposited, bringing the balance to ₹5000.

- **Withdraw:** ₹6000 is withdrawn, leaving the balance at ₹-1000 (within the overdraft limit of ₹10,000).
- **Withdraw again:** ₹8000 is withdrawn, but the total amount to be withdrawn exceeds the overdraft limit (₹5000 balance + ₹10,000 overdraft limit = ₹15,000). Since ₹8000 is within the overdraft limit, the withdrawal is allowed.
- Finally, the account info is logged with the updated balance.

This code effectively simulates a simple banking system with savings and current accounts, showcasing inheritance, method overriding, and managing different behaviors for different account types.

4o mini

You said:

hindi me

ChatGPT said:

बिल्कुल! चलिए इस कोड को हिंदी में समझते हैं, लाइन दर लाइन।

Abstract Base Class: **BankAccount**

javascript

CopyEdit

```
class BankAccount {
  constructor(accountNumber, owner, balance = 0) {
    if (new.target === BankAccount) {
      throw new Error("Cannot instantiate abstract class directly");
    }
    this.accountNumber = accountNumber;
    this.owner = owner;
    this.balance = balance;
  }
}
```

- **BankAccount क्लास:** यह एक एब्सट्रैक्ट क्लास है, जिसे डायरेक्टली इंस्टेंटिएट (instantiate) नहीं किया जा सकता। अगर आप इस क्लास का डायरेक्ट उदाहरण बनाने की कोशिश करेंगे तो यह एक एरर फेंक देगा।
- **कंस्ट्रक्टर (constructor):** यह `accountNumber`, `owner`, और `balance` प्रॉपर्टीज़ को इनिशियलाइज़ करता है। अगर `BankAccount` क्लास को डायरेक्टली इनिशियलाइज़ किया जाए, तो एरर थ्रो करता है।

javascript

CopyEdit

```
deposit(amount) {
    throw new Error("Method 'deposit()' must be implemented in subclass");
}

withdraw(amount) {
    throw new Error("Method 'withdraw()' must be implemented in subclass");
}
```

- **एब्सट्रैक्ट मेथड्स (deposit और withdraw):** इन मेथड्स का डीफॉल्ट इम्प्लीमेंटेशन नहीं है। इन्हें सबक्लास में इम्प्लीमेंट करना होगा, और अगर सबक्लास में इनका इम्प्लीमेंटेशन नहीं होता, तो एरर आएगा।

javascript

CopyEdit

```
getBalance() {
    return this.balance;
}

getAccountInfo() {
    return {
        accountNumber: this.accountNumber,
        owner: this.owner,
        balance: this.balance
    };
}
}
```

- **getBalance मेथड:** यह खाता (account) का बैलेंस रिटर्न करता है।
- **getAccountInfo मेथड:** यह खाता की सारी जानकारी (accountNumber, owner, और balance) एक ऑब्जेक्ट के रूप में रिटर्न करता है।

SavingsAccount क्लास (जो BankAccount का सबक्लास है)

javascript

CopyEdit

```
class SavingsAccount extends BankAccount {
  constructor(accountNumber, owner, balance = 0) {
    super(accountNumber, owner, balance);
    this.interestRate = 0.03; // 3% ब्याज
  }
}
```

- **SavingsAccount क्लास:** यह `BankAccount` से विरासत में मिली है और इसमें बचत खाते (savings account) के लिए विशेषताएँ हैं।
- **कंस्ट्रक्टर (constructor):** इसमें `interestRate` (3% ब्याज दर) को इनिशियलाइज किया गया है, साथ ही `accountNumber`, `owner`, और `balance` को सुपर क्लास से प्राप्त किया गया है।

javascript

CopyEdit

```
deposit(amount) {
  if (amount > 0) {
    this.balance += amount;
    console.log(`₹${amount} Savings में जमा किया गया। बैलेंस:
₹${this.balance}`);
  }
}
```

- **deposit मेथड:** यदि जमा की जाने वाली राशि (`amount`) सकारात्मक है (0 से बड़ी), तो वह राशि `balance` में जोड़ दी जाती है और जमा करने की जानकारी लॉग में दिखती है।

javascript

CopyEdit

```
withdraw(amount) {
  if (amount <= this.balance) {
    this.balance -= amount;
    console.log(`₹${amount} Savings से निकाला गया। बैलेंस:
₹${this.balance}`);
  } else {
    console.log("Savings में पर्याप्त बैलेंस नहीं है।");
  }
}
```


- **withdraw मेथड:** यदि निकासी राशि (withdrawal amount) बैलेंस से कम या बराबर है, तो वह राशि **balance** से घटा दी जाती है। अगर बैलेंस कम है, तो "Insufficient balance" (पर्याप्त बैलेंस नहीं है) संदेश दिखाया जाता है।

javascript

CopyEdit

```
applyInterest() {
  const interest = this.balance * this.interestRate;
  this.balance += interest;
  console.log(`₹${interest.toFixed(2)} ब्याज जोड़ा गया। नया बैलेंस:
₹${this.balance}`);
}
```

- **applyInterest मेथड:** यह ब्याज की राशि (interest) निकालता है, जो बैलेंस का 3% होती है, और उसे बैलेंस में जोड़ देता है। फिर, नए बैलेंस के साथ ब्याज की जानकारी लॉग में दिखाई जाती है।

CurrentAccount क्लास (जो BankAccount का सबक्लास है)

javascript

CopyEdit

```
class CurrentAccount extends BankAccount {
  constructor(accountNumber, owner, balance = 0) {
    super(accountNumber, owner, balance);
    this.overdraftLimit = 10000; // ओवरड्राफ्ट लिमिट
  }
}
```

- **CurrentAccount क्लास:** यह **BankAccount** से विरासत में मिली है और इसमें चालू खातों (current accounts) के लिए अतिरिक्त विशेषताएँ हैं।
- **कंस्ट्रक्टर (constructor):** इसमें **overdraftLimit** को ₹10,000 सेट किया गया है, जो एक लिमिट होती है, जिसे ग्राहक अपने खाता बैलेंस से ऊपर उधारी (overdraft) के रूप में निकाल सकता है।

javascript

CopyEdit

```
deposit(amount) {
  if (amount > 0) {
    this.balance += amount;
  }
}
```

```
        console.log(`₹${amount} Current में जमा किया गया। बैलेंस: ₹${this.balance}`);
    }
}
```

- **deposit** **मैथड**: अगर जमा की गई राशि (amount) सही है (0 से अधिक), तो वह राशि **balance** में जोड़ दी जाती है और जमा की जानकारी लॉग में दिखाई जाती है।

javascript

CopyEdit

```
withdraw(amount) {
    if (amount <= this.balance + this.overdraftLimit) {
        this.balance -= amount;
        console.log(`₹${amount} Current से निकाला गया। बैलेंस: ₹${this.balance}`);
    } else {
        console.log("ओवरड्राफ्ट लिमिट से अधिक निकासी।");
    }
}
```

- **withdraw** **मैथड**: यदि निकासी राशि (withdrawal amount) खाता बैलेंस और ओवरड्राफ्ट लिमिट के संयुक्त रूप से कम या बराबर है, तो राशि **balance** से घटा दी जाती है। अगर राशि अधिक है, तो "ओवरड्राफ्ट लिमिट से अधिक निकासी" संदेश दिखाया जाता है।

कोड का परीक्षण (Testing the Code)

javascript

CopyEdit

```
const savings = new SavingsAccount("111", "Aman", 5000);
savings.deposit(1000);
savings.withdraw(2000);
savings.applyInterest();
console.log("Savings Account Info:", savings.getAccountInfo());
```

- **SavingsAccount उदाहरण**: खाता नंबर 111, मालिक Aman, और बैलेंस ₹5000 के साथ एक **SavingsAccount** उदाहरण बनाया गया है।
- **जमा (Deposit)**: ₹1000 जमा किया गया, जिससे बैलेंस ₹6000 हो गया।

- **निकासी (Withdrawal):** ₹2000 निकाला गया, जिससे बैलेंस ₹4000 हो गया।
- **ब्याज (Interest):** ब्याज 3% (₹4000 का 3%) के हिसाब से ₹120 जोड़ा गया, जिससे बैलेंस ₹4120 हो गया।
- अंत में, खाता जानकारी (Account Info) लॉग में दिखाई जाती है।

javascript

CopyEdit

```
const current = new CurrentAccount("222", "Neha", 2000);
current.deposit(3000);
current.withdraw(6000);
current.withdraw(8000); // ओवरड्राफ्ट लिमिट से बाहर निकासी
console.log("Current Account Info:", current.getAccountInfo());
```

- **CurrentAccount उदाहरण:** खाता नंबर 222, मालिक Neha, और बैलेंस ₹2000 के साथ एक CurrentAccount उदाहरण बनाया गया है।
- **जमा (Deposit):** ₹3000 जमा किया गया, जिससे बैलेंस ₹5000 हो गया।
- **निकासी (Withdrawal):** ₹6000 निकाला गया, जिससे बैलेंस ₹-1000 (ओवरड्राफ्ट लिमिट में) हो गया।
- **निकासी (Withdrawal):** ₹8000 निकाला गया, जो ओवरड्राफ्ट लिमिट के तहत वैध था, क्योंकि ₹5000 बैलेंस + ₹10000 ओवरड्राफ्ट लिमिट = ₹15000 है।